Information Disclosure Statement

1. The information disclosure statement filed 7/13/2006 fails to comply with 37 CFR 1.98(a)(2), which requires a legible copy of each cited foreign patent document; each non-patent literature publication or that portion which caused it to be listed; and all other information or that portion which caused it to be listed. Specifically, a copy of the foreign patent DE 41 12 090 A1 and a copy of the NPL reference James Stichnoth, "Generating Code for High-Level Operations through code Composition" were not included. Examiner was able to retrieve a copy of Stichnoth, therefore, Stichnoth has been considered. However, foreign patent DE 41 12 090 A1 is not considered because the information disclosure statement filed does not include a copy of the foreign patent which is not in the English language and does not include a concise explanation of its relevance, as it is presently understood by the individual designated in 37 CFR 1.56(c) most knowledgeable about the content of the information.

EXAMINER'S AMENDMENT

2. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

3. Authorization for this examiner's amendment was given in a telephone interview with George Hartnell III (Reg No. 42,639) on February 18, 2011.

4. The specification has been amended as follows:

a. Page 16, line 26, delete "level 2" and insert -- level 1 --.

b. Claim 17, line 7, delete "level 1" and insert -- level 2 --.

5. The claims have been amended as follows:

a. Claim 1: replace claim 1 with the following:

1. A system for operating automatically on source code submitted by users to generate optimized code suitable for running on a predefined hardware platform comprising at least one processor, and for use in a predetermined field of application, the system being characterized in that it comprises means for receiving symbolic code sequences referred to as benchmark sequences representative of the behavior of said processor in terms of performance, for the predetermined field of application; means for receiving first static parameters defined on the basis of the predefined hardware platform, its processor, and the benchmark sequences; means for receiving dynamic parameters also defined on the basis of the predefined hardware platform, its processor, and the benchmark sequences; an analyzer device for establishing optimization rules from tests and measurements of performance carried out using the benchmark sequences, the static parameters, and the dynamic parameters; a device for optimizing and generating code receiving firstly the benchmark sequences and secondly the optimization rules for examining the user source code, detecting optimizable loops, decomposing them into kernels, and assembling and injecting code to deliver the optimized code; and means for reinjecting information coming from the device for generating and optimizing code and

relating to the kernels back into the benchmark sequences, wherein the benchmark sequences comprise a set of simple and generic loop type code fragments specified in a source type language and organized in a hierarchy of levels by order of increasing complexity of the code for the loop body.

b. Claim 8, line 7, delete "utilitzation" and insert -- utilization --.

c. Claim 16 is cancelled.

d. Claim 17, line 1, delete "claim 16" and insert -- claim 1 --.

e. Claim 19, line 3, delete "level 1" and insert -- level 2 --.

f. Claim 20, line 1, delete "claim 16" and insert -- claim 1 --.

g. Claim 21, line 1, delete "claim 16" and insert -- claim 1 --.

h. Claim 29 is cancelled.

i. Claim 30, line 1, delete "claim 29" and insert -- claim 28 --.

j. Claim 30, line 15, delete "level 1" and insert -- level 2 --.

Reasons for Allowance

6. The following is an examiner's statement of reasons for allowance: the prior art of record fails to teach or suggest the claimed invention. Specifically, the prior art of record fails to teach a device for optimizing and generating code receiving firstly the benchmark sequences, means for reinjecting information coming from the device for generating and optimizing code and relating to the kernels back into the benchmark sequences, wherein the benchmark sequences comprises a set of simple and generic loop type code fragments specified in source type language and organized in a hierarchy of levels by order of increasing complexity of the code for the loop body.

7. The closest prior art of record, McCollum teaches a system for operating automatically on source code submitted by users to generate optimized code suitable for running on a predefined hardware platform comprising at least one processor, and for use in a predetermined field of application, the system being characterized in that it comprises means for receiving symbolic code sequences referred to as benchmark sequences representative of the behavior of said processor in terms of performance, for the predetermined field of application (i.e., a representative selection of loops for training the neural network, see page 696, paragraph 2, pages 698-699, section 8.2); means for receiving first static parameters defined on the basis of the predefined hardware platform, its processor, and the benchmark sequences (i.e., the number of processors available, see page 699, paragraph 2); means for receiving dynamic parameters also defined on the basis of the predefined hardware platform, its processor, and the benchmark sequences (i.e., characteristics that are considered to be important in the choice of an appropriate

data partitioning scheme, see page 696, last paragraph, page 699, paragraph 2); an analyzer device for establishing optimization rules from tests and measurements of performance carried out using the benchmark sequences, the static parameters, and the dynamic parameters (i.e., determining a suitable data distribution strategy, see page 695, paragraph 3); a device for optimizing and generating code receiving secondly the optimization rules for examining the user source code, detecting optimizable loops, decomposing them into kernels, and assembling and injecting code to deliver the optimized code (i.e., generating actual parallel code based on the determined data distribution strategy, see page 695, paragraph 3), wherein the benchmark sequences comprises a set of simple and generic loop type code fragments specified in source type language and organized in a hierarchy of levels by order of increasing complexity of the code for the loop body (i.e., training cases reflect the complexities found in real codes, example codes have been identified at levels 1, 2, and 3 within the loop corpus, see page 697-698, section 7, page 698, paragraph 4).

However, McCollum fail to teach the independent claim as recited where the benchmark sequences comprises a set of simple and generic loop type code fragments specified in source type language and organized in a hierarchy of levels by order of increasing complexity of the code for the loop body are received as input to the device for optimizing and generating code and information coming from the device for generating and optimizing code and relating to the kernels are reinjected back into the benchmark sequences.

8. Any comment considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue

fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

Conclusion

9. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

- Wada et al. (US 2003/0110481 A1) is cited to teach a method of detecting a set of compiler directives to be applied to program portions through measuring the execution time of each compiler directive and selecting the compiler directives with the shortest execution time.
- Atkin et al. (US 2006/0005177 A1) is cited to teach a method determining an optimized JVM for executing an application through data mining.
- Callahan et al. "Vectorizing Compilers: A Test Suite and Results", 1988, Proceedings of the 1988 ACM/IEEE conference on Supercomputing.

  This document is cited to teach a test suite of loops of different complexity to test a vectorizing compiler.
- McCollum et al. "A Meta-heuristic Approach to Parallel Code Generation", 2002, Proceedings of the 5th international conference on High performance computing for computational science.
- Monsifrot et al. "A machine Learning Approach to Automatic Production of Compiler Heuristics", 2002, Proceedings of the 10th International Conference on Artificial Intelligence: Methodology, Systems, and Applications.

This document is cited to teach automatic generation of optimization heuristics for a target processor by machine learning.

- Stephenson et al. "Meta Optimization: Improving Compiler Heuristics with Machine Learning", 2003, Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation.

  This document is cited to teach a method for automatically fine-tuning compiler heuristics by searching the space of compiler heuristics using a training set.

- Che et al. "Optimization Parameter Selection by Means of Limited Execution and Genetic Algorithms", 2003, Lecture Notes in Computer Science, Volume 2834.

  This document is cited to teach searching for optimal optimization parameter vectors for a compiler based on a program's real execution time.

- None of these cited references teach the independent claim as recited where the benchmark sequences comprises a set of simple and generic loop type code fragments specified in source type language and organized in a hierarchy of levels by order of increasing complexity of the code for the loop body are received as input to the device for optimizing and generating code and information coming from the device for generating and optimizing code and relating to the kernels are reinjected back into the benchmark sequences.

10. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Jue S. Wang whose telephone number is (571) 270-1655. The examiner can normally be reached on M-F 9:30 am - 5:00pm (EST).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis Bullock can be reached on 571-272-3759. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Lewis A. Bullock, Jr./
Supervisory Patent Examiner, Art Unit 2193

/Jue S Wang/
Examiner, Art Unit 2193